2D Distributed Label Propagation on 400 GPUs[1]

George M. Slota, Michael Mandulak, Ujwal Pandey, and Anthony Fabius
Rennselaer Polytechnic Institute

IEEE High Performance Extreme Computing Conference (HPEC)

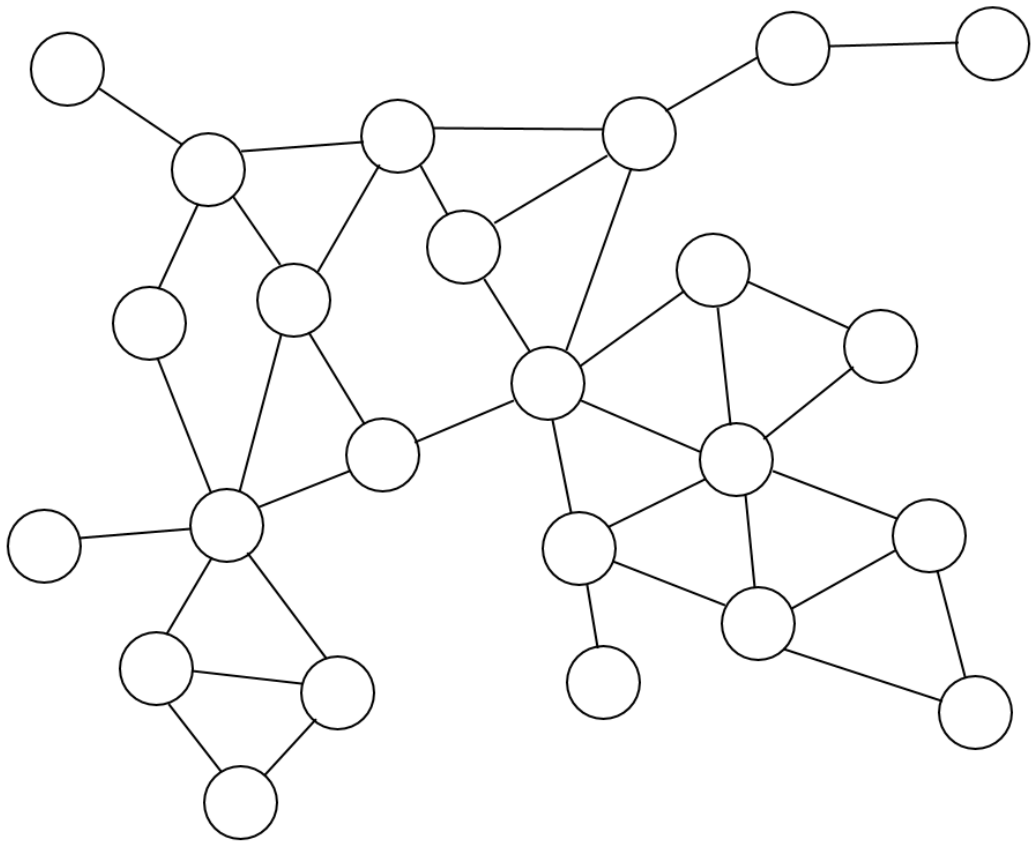
September 16th, 2025

[1]This work is supported by the National Science Foundation under Grant No. 2047821.

The Label Propagation Algorithm

for community detection

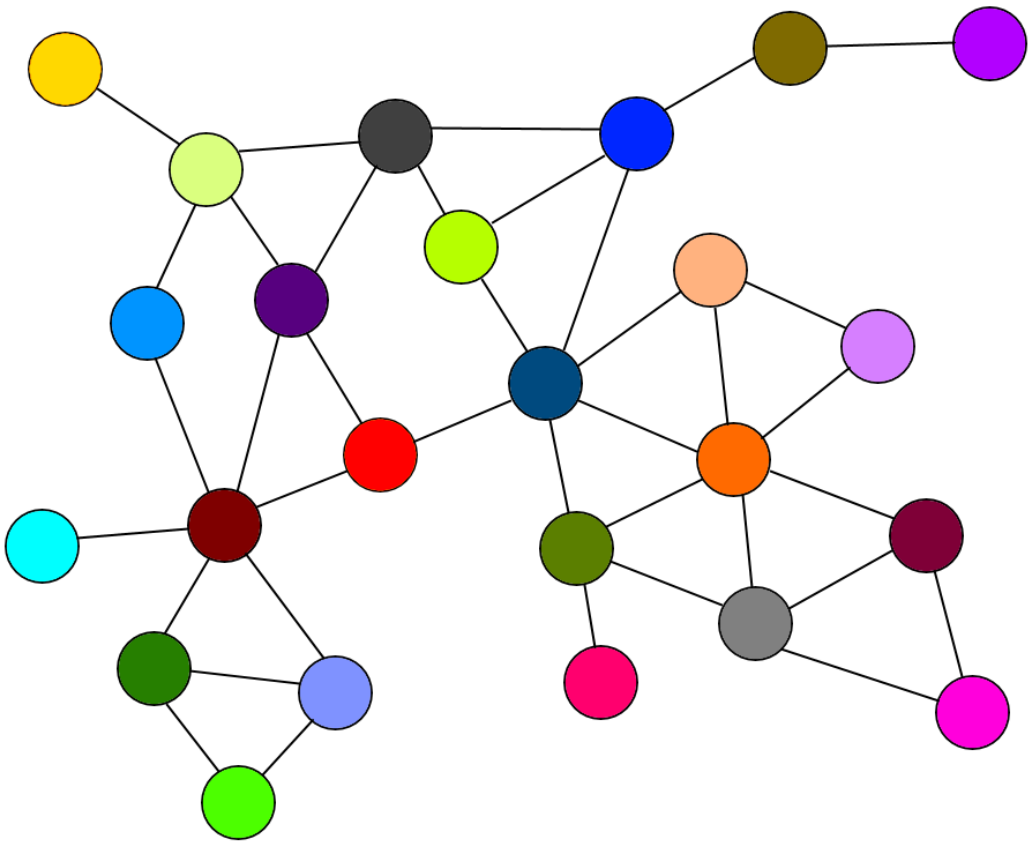
- Consider an input graph. Initialize every vertex to a unique label.



The Label Propagation Algorithm

for community detection

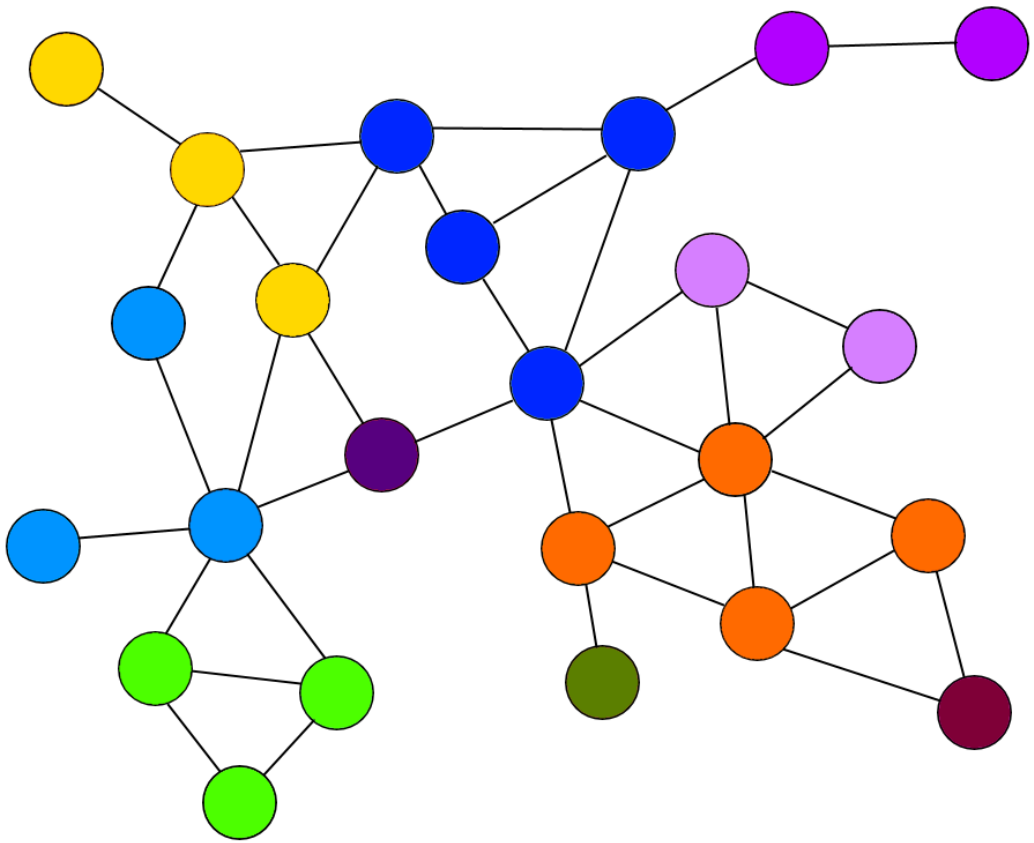
- Consider an input graph. Initialize every vertex to a unique label.



The Label Propagation Algorithm

for community detection

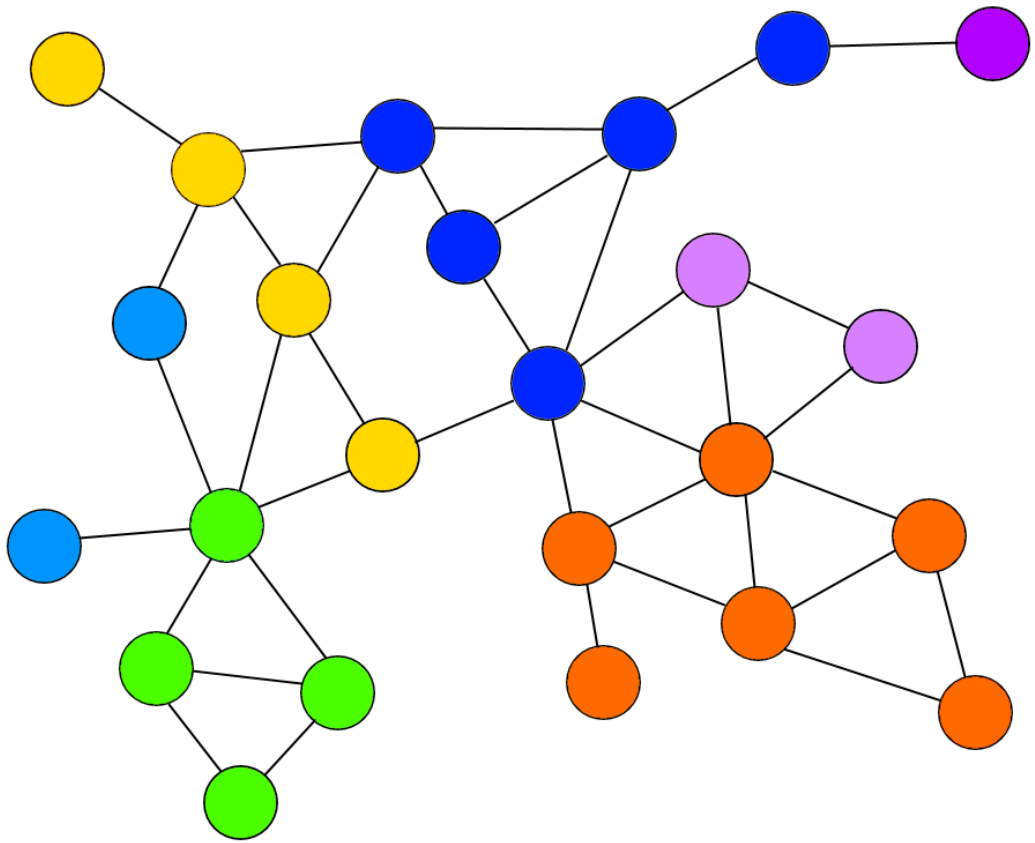
- Consider an input graph. Initialize every vertex to a unique label.
- Every vertex assumes the *statistical mode* of labels in its neighborhood, with ties broken randomly.



The Label Propagation Algorithm

for community detection

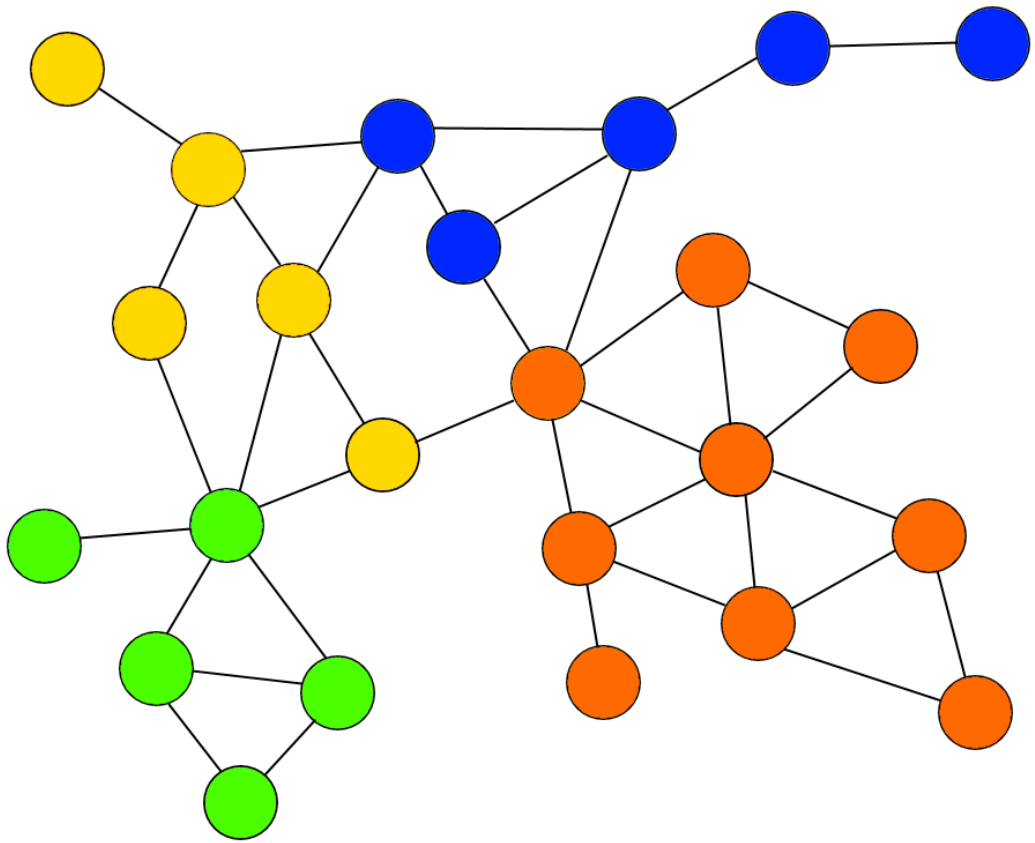
- Consider an input graph. Initialize every vertex to a unique label.
- Every vertex assumes the *statistical mode* of labels in its neighborhood, with ties broken randomly.
- Iterate until convergence or for a fixed iteration count.



The Label Propagation Algorithm

for community detection

- Consider an input graph. Initialize every vertex to a unique label.
- Every vertex assumes the *statistical mode* of labels in its neighborhood, with ties broken randomly.
- Iterate until convergence or for a fixed iteration count.



Label Propagation Applications

Label propagation is widely used across multiple domains.

Community Detection:

- Often used directly or with modification (multiple labels per vertex, more advanced reduction than mode, etc.)

Machine Learning:

- Can be used to label missing data, for generalized clustering, and related problems.

Graph Partitioning:

- Used for both coarsening as well as direct cut minimization and balance.

Because of the reliance on hash tables for statistical mode calculations, high performance implementations of Label Propagation are deceptively challenging.

The Contributions of This Work

We present the first multi-node distributed GPU implementation of the Label Propagation algorithm.

We are also the first to explicitly use a 2D graph distribution for the label propagation problem.

In addition, we significantly improve upon prior art...

Our Work Relative to Prior Art

Input Scale - largest input processed on GPUs:

- Prior Work: 3.3 billion edge uk-2007-05 graph.
- Us: 128 billion edge Web Data Commons 2012 (WDC12).
- **Improvement:** 39× larger.

System Scale - number of GPUs utilized:

- Prior Work: 2 GPUs on a single node.
- Us: 400 GPUs across 67 nodes.
- **Improvement:** 200× more GPUs.

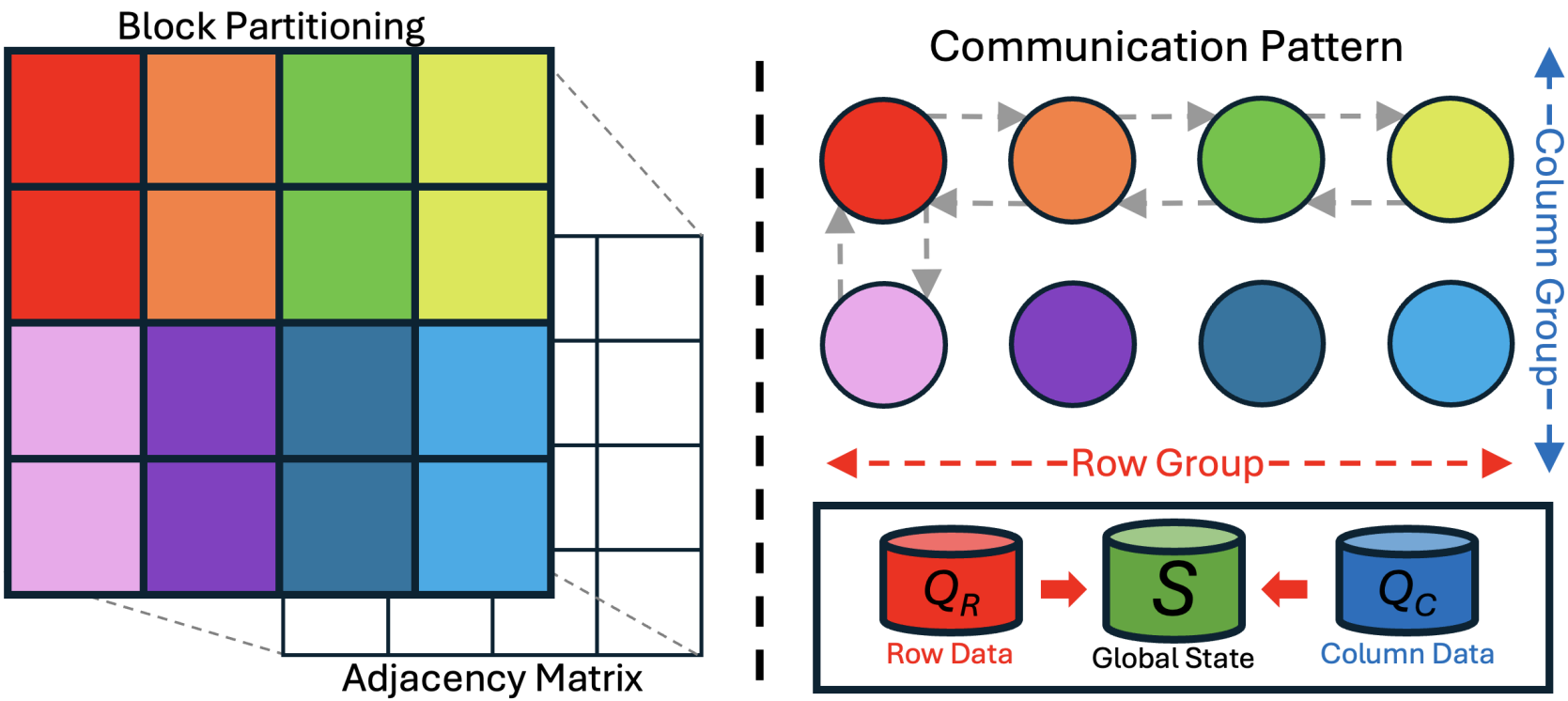
Speed at Scale - fastest known times for WDC12:

- Prior Work: 40 seconds per iteration.
- Us: 10 seconds per iteration.
- **Improvement:** 4× faster.

Note: At the small scale, we likely aren't competitive with any prior work.

2D Graph Distributions

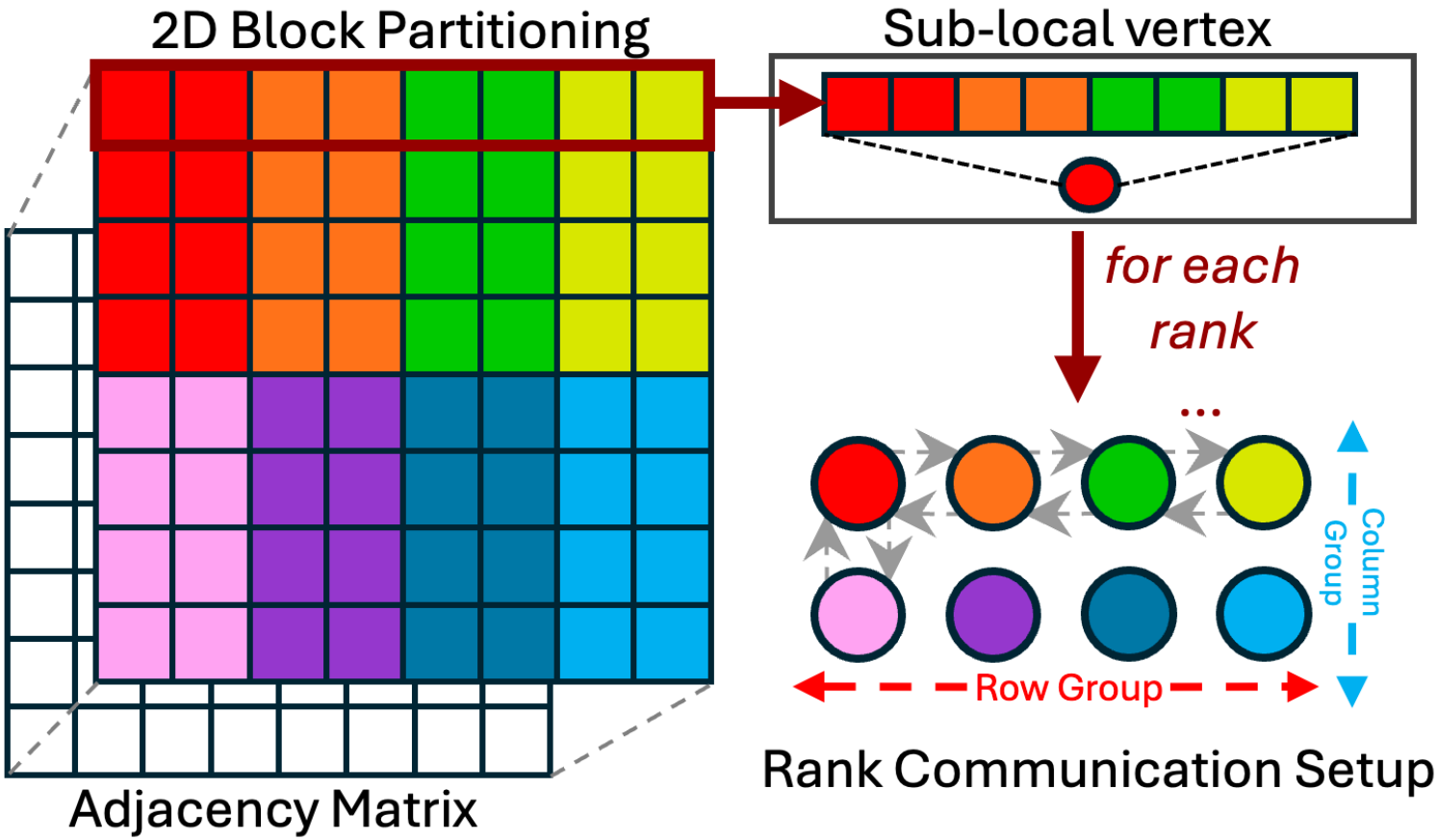
Adjacency matrix is distributed into 2D blocks. Edge-based propagation of graph data occurs in two phases, along *row groups* and then *column groups*. This reduces number of messages for large-scale multi-hundred rank runs.



2.5D Graph Distributions

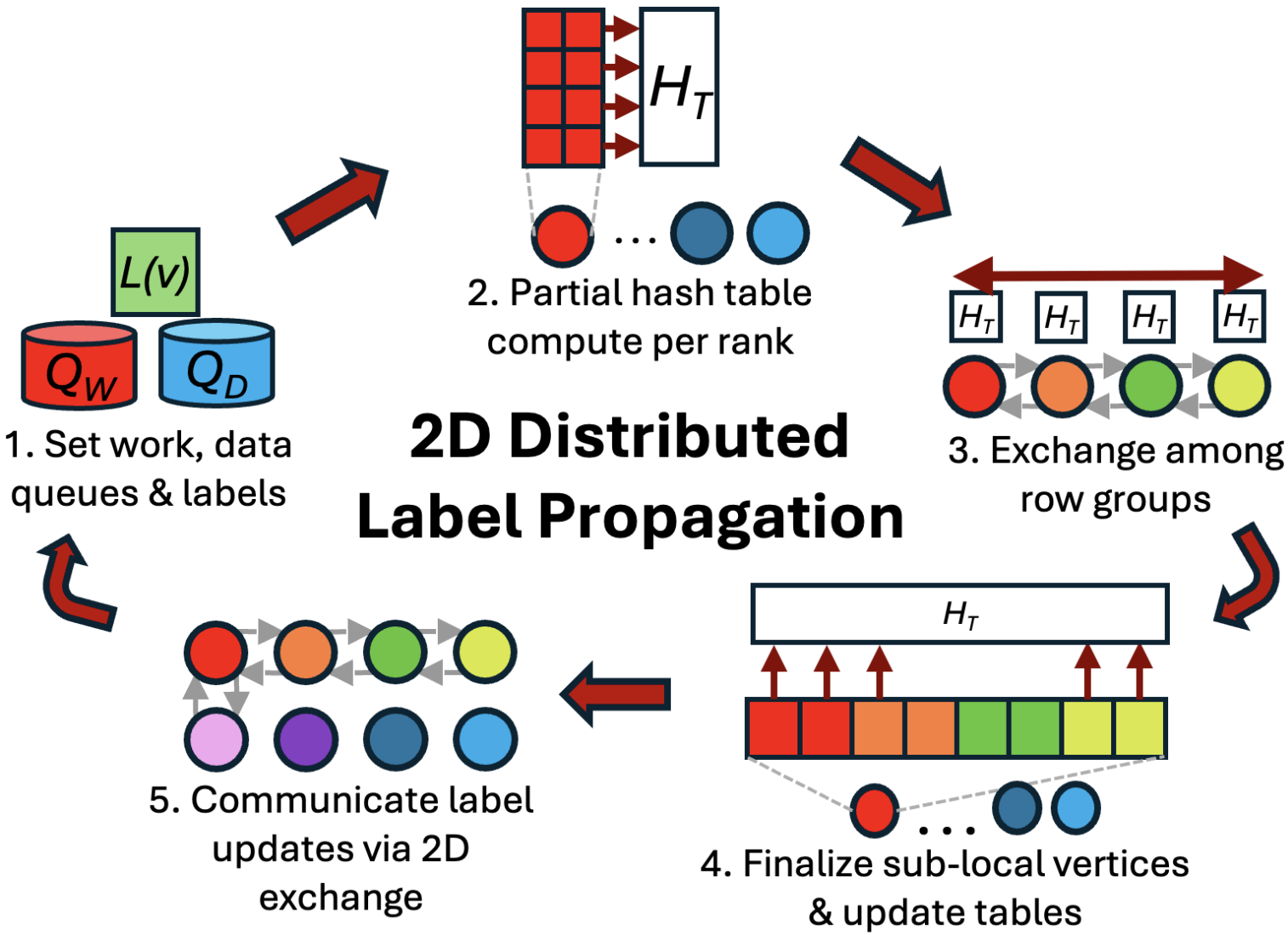
A new level in the hierarchy

As a statistical mode computation requires a hash table and is nontrivial within a basic 2D comm. pattern, we further consider a sub-local partitioning of row group-owned vertices.



Overview of Our Algorithm

Using distributed hash tables and 2.5D communications



Implementation Details

Code: All of our code is implemented in C++ with CUDA, NCCL, and MPI. We will publicly release our code to `https://github.com/HPCGraphAnalysis/HPCGraph`.

Graph Structure: We use the 2D graph data structure and communication backend developed in our prior work – presented last week at ICPP25. We require no external library dependencies beyond this and the above.

Hash Table: A key aspect for performance. We consider linear, quadratic, and double probing methods. We set the capacity for each vertex's table to be equal in size to the number of insertions (not an issue like you might think).

Label Initialization: Randomly select label of neighbor. Equivalent to what occurs in iteration-synchronized execution.

Experimental Setup

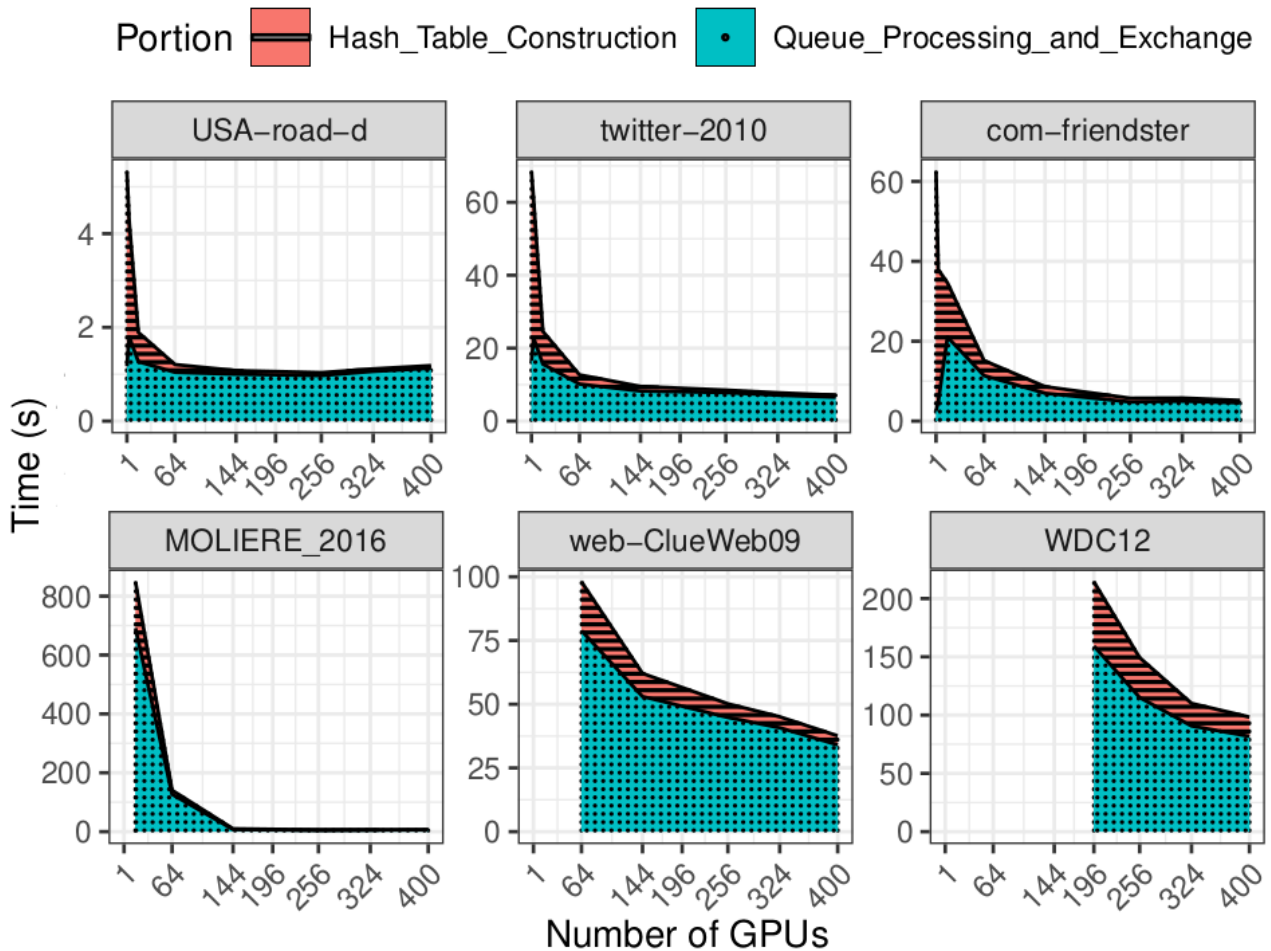
Test graphs and test system

Name	Vertices	NNZ	d_{avg}
USA-road-d	24 M	116 M	5
twitter-2010	41 M	2.9 B	72
com-friendster	65 M	3.6 B	55
MOLIERE_2016	30 M	6.7 B	223
web-ClueWeb09	1.7 B	16 B	9
gsh-2015	1.0 B	66 B	66
WDC12	3.6 B	257 B	73

Test System: RPI's AiMOS system at the CCI. We use up to 400 GPUs on 67 nodes, each with 2× IBM Power-9 CPUs and 512 GB DRAM, 6× NVIDIA 32 GB V100 GPUs, connected via an EDR Infiniband network.

Scalability

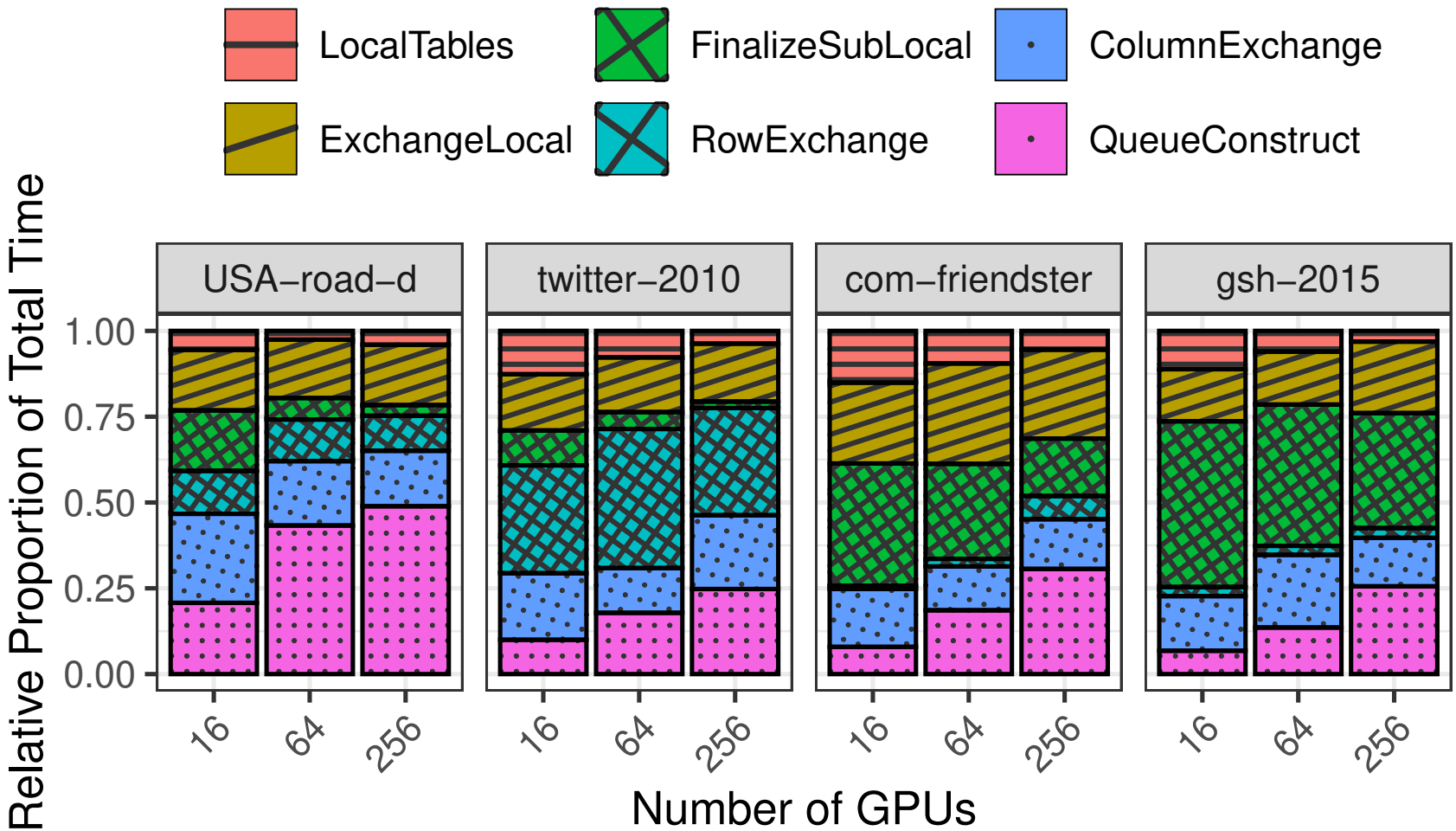
Strong Scaling up to 400 GPUs



All inputs strong scale to 400 GPUs except for the smallest road network, which can fit on a single GPU replicated about 20×.

Algorithm Timing Breakdown

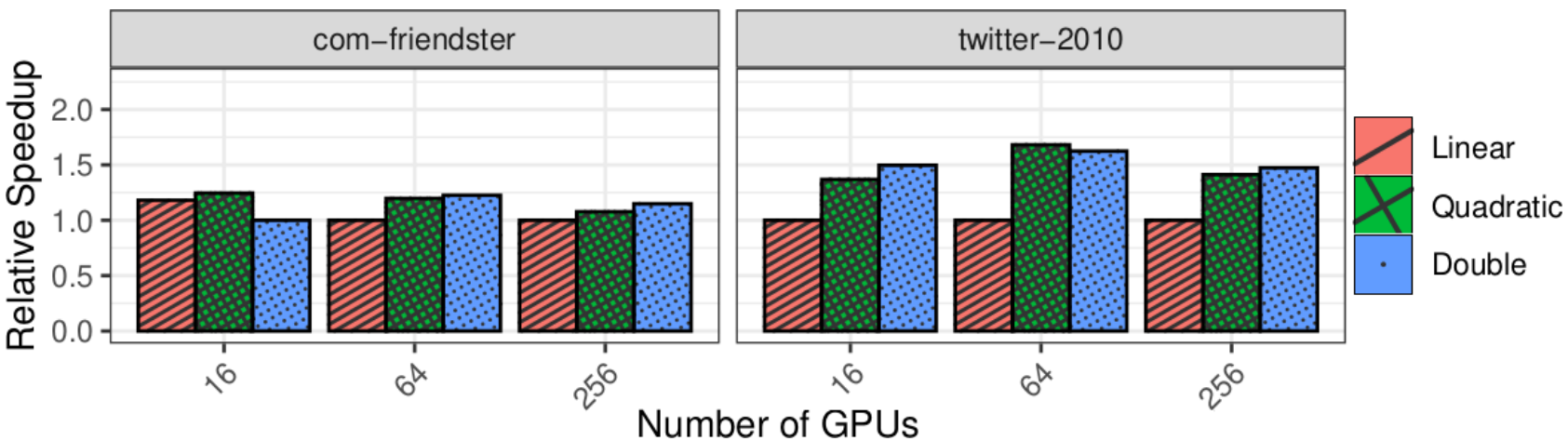
At the large scale, table computations become almost irrelevant.



Communication tends to dominate timings with a large number of concurrent GPUs, as expected.

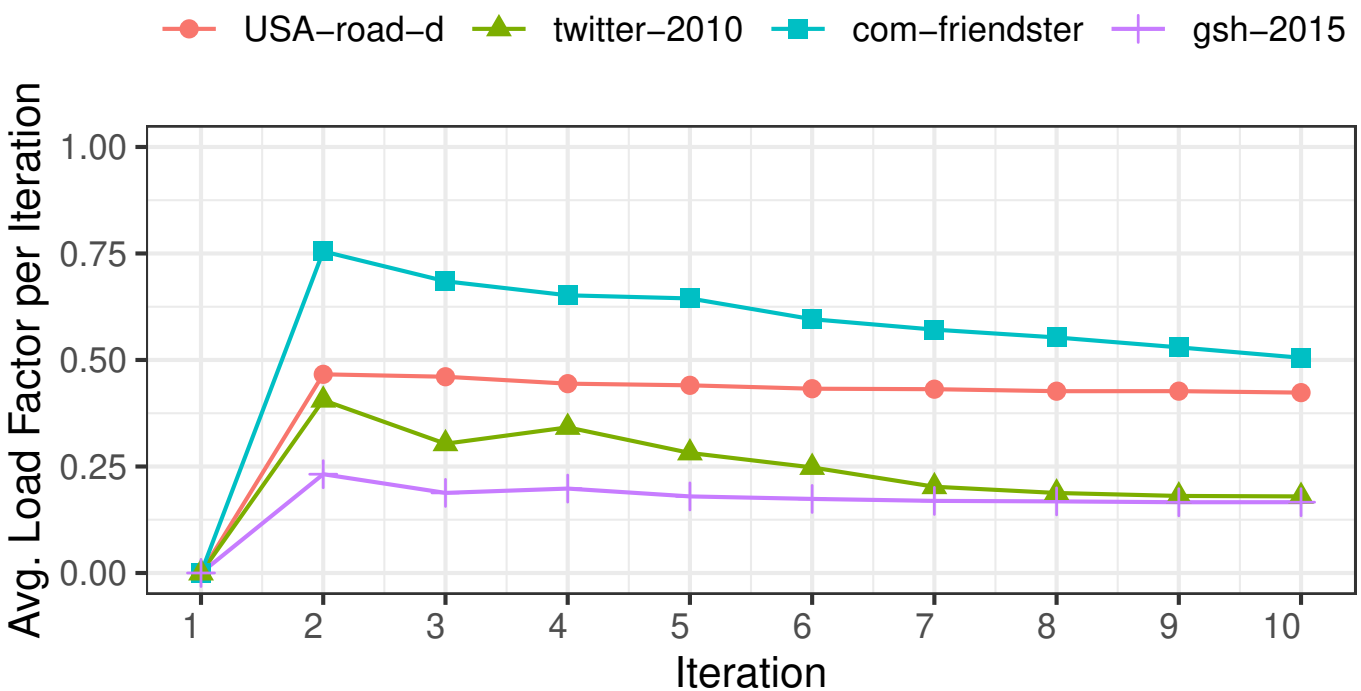
Hash Table Performance

- On average, quadratic double probing runs fastest, mirroring prior work.



Hash Table Performance

- On average, quadratic double probing runs fastest, mirroring prior work.
- Load factor is never an issue with minimum capacity allocation, even on the first iterations.



Conclusions

and Thanks!

We present a 2D distributed GPU label propagation algorithm, which scales to the largest existing datasets and 400 GPUs.

- We significantly improve over prior art for the problem, in all of input scale, distributed concurrency, and solve time.
- We note that local table computations become almost irrelevant at the large scale.
- **Future Work**:
  - Further optimize 2.5D approach and communication routines, investigate asynchronous methods.

Contact: `gmslota@gmail.com`, `www.gmslota.com`